**CS 260 Self Evaluation for Assignment 6 – Recursive Sorts**

|  |  |
| --- | --- |
| Your name:  David Oftedahl | Date:  12/04/2022 |
| Are you willing to allow your code to be used in example debugging demonstrations or documentation?  Yes  No | |

**Instructions – Part 1**  
This document is to be turned in alongside solution of this lab. You will use this document to indicate your status on the lab, as well as areas where you are struggling conceptually or in converting concept to code. Please use the space underneath each evaluation criteria to describe any errors you are receiving or challenges you are having implementing the required functionality for your code.

**Project Organization (please fill in for your language)**

|  |  |
| --- | --- |
| ***File Structure (C++)*** | |
| Is your class split into a header and a source file? |  |
|  | |
|  | |
| ***File Structure (C#)*** | |
| Is your class declared in a class library, separate from your console driver? |  |
|  | |
|  | |
| ***File Structure (Python)*** | |
| Is your class declared in a separate, imported file? |  |
| Yes | |

**Functionality**

|  |  |
| --- | --- |
| **Base Lab** | |
|  | |
| ***Heap*** | |
| Does the program compile without errors or warnings and run without crashing? |  |
| Yes, Heap complies without issues | |
| Does the class properly add and remove items? |  |
| Yes, Heap is able to add and remove items. | |
| Does it meet the test examples properly? |  |
| Yes, the test examples are met properly | |
| ***Priority Queue*** | |
| Does the program compile without errors or warnings and run without crashing? |  |
| Yes, it complies without errors or warnings | |
| Is it built as a wrapper (composition) upon a heap instance? |  |
| Yes, it was built on top of Heap instance. | |
| Does the class properly add and remove items? |  |
| Yes, it properly removes items. | |
| **Advanced Lab** | |
|  | |
| ***HeapSort*** | |
| Does the program compile and run without errors or warnings and run without crashing? |  |
| Yes, it complies and runs without error. | |
| Does it properly sort the array from smallest to largest? |  |
| Yes, it sorts from smallest to largest | |
| Does it work by heapifying the data and then taking it away one item at a time? |  |
| Yes, it uses a heapfy and takes away the items one at a time. | |

|  |  |
| --- | --- |
| ***MergeSort*** | |
| Does the program compile without errors or warnings and run without crashing? |  |
| Yes, it compiles and runs without issue. | |
| Does it properly sort the array from smallest to largest? |  |
| Yes, it sorts the array from smallest to largest | |
| Does it work by recursively breaking the array into halves and then merging them? |  |
| Yes, it uses recursion the break the array into halves over and over. | |
| ***QuickSort*** | |
| Does the program compile without errors or warnings and run without crashing? |  |
| Yes, it compiles and runs without error. | |
| Does it properly sort the array from smallest to largest? |  |
| Yes, it sorts from smallest to largest. | |
| Does it work by selecting a pivot and sorting partitioned values relative the pivot value? |  |
| Yes, it selects a pivot point and then sorts partitioned values from there. | |
| **Thinking Problem** | |
|  | |
| ***FindNth*** | |
| Does the program compile without errors or warnings and run without crashing? |  |
|  | |
| Does it properly stop sorting once the nth value has been found? |  |
|  | |

**Instructions – Part 2**   
Please answer the following questions, in your own words, regarding your experiences throughout this lab.

**Experiential Review**

|  |
| --- |
| **What aspects of this lab did you find most challenging?** |
| I ran into some significant issues with the mergeSort but was able to determine the issue I had was caused by the parameter array never getting updated. I could see through the extra printing I added throughout that the splitting and sorting was working, but couldn’t tell why the end results kept resorting back to the original unsorted arrays. |
| **What concept from this lab do you feel you have the best grasp on now?** |
| While it took me a little while to figure it out at first, I feel pretty confident about creating Heaps and how to use them for a priority queue. |
| **Please summarize the basic concepts of how each of these sorts functions and what the pros and cons are for each:** |
| HeapSort: Turns an array into a heap, then uses the structure of the heap to pull them off in order. Beneficial because it doesn’t require extra memory for additional arrays, but it is a O(log N) for both adding and removing.  MergeSort: Recursively splits the array into sub-arrays until it reaches length 1, then combines the back together after sorting each side. Requires extra memory for the sub-arrays, but benefits from being a stable sort.  QuickSort: After picking a starting value, it recurses through the array moving items until the pivot selected is in it’s proper place, and all items before and after it are on the right side. Can have worst case performance of N^2, but doesn’t require extra memory. |
| **When might you choose to use an insertion or selection sort instead of one of the recursive sorts covered this week?** |
| An insertion or selection sort would be better for smaller amounts of data, because they are easier to manage. Larger amounts of data benefit from recursive sorts because they have potential to greatly cut down on the time required. |